Curso de programación en Android

20/Junio/2012

Ramón Alcarria
Augusto Morales

Manejo del teléfono

Permisos
Servicios del sistema
    Creando un servicio local
Acceso a Sensores
Acceso a la Cámara de Fotos
Acceso a información de la tarjeta WiFi
Posicionamiento / Geocodificación
Manejo de Notificaciones

Permisos

Para establecer un permiso para una aplicación, es necesario declarar uno o más elementos <uses-permission>, donde se especifica el tipo de permiso que se desea habilitar.

Por ejemplo, si se desea que una aplicación pueda monitorizar mensajes SMS entrantes, tendríamos que introducir algo como esto en el fichero AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  android:versionCode="1" android:versionName="1.0" package="com.marakana.yamba1">
  <application android:icon="@drawable/icon" android:label="@string/app_name">

    <activity android:name=".StatusActivity" android:label="@string/titleStatus">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

  </application>
  <uses-sdk android:minSdkVersion="4" />

  <uses-permission android:name="android.permission.INTERNET" /> <!-- ❶ -->
</manifest>
```

Permisos

Para todos los permisos disponibles ver la clase Android.Manifest.permission:

http://developer.android.com/reference/android/Manifest.permission.html

Servicios del sistema

Ya hemos dicho que la mayoría de capacidades del terminal móvil: Localización, Sensores, WiFi, Alarma, Telefonía, Bluetooth, etc se utilizan a través de servicios, que se ejecutan en segundo plano.

Servicios locales Servicios que forman parte de la aplicación. Servicios **privados** solo accesibles desde su aplicación

Servicio remotos Servicios que forman parte de otras aplicaciones instaladas en el dispositivo y están accesibles de forma **pública** por otras aplicaciones

Existen dos roles cuando se trabaja con servicios

Controlador del servicio Es el encargado de arrancar y parar el servicio. Este rol lo puede cumplir cualquier componente

Cliente del servicio Se conecta al servicio obteniendo una referencia de su IBinder, desde el que podrá hacer peticiones al mismo

Creando un servicio local

1. Crea una clase que extienda la clase **Service**
2. Sobreescribe los métodos onCreate, onDestroy
3. Implementa la tarea que va a llevar a cabo el servicio
4. Sobreescribe el método onBind() si se va a permitir peticiones de un cliente
5. Crea la interfaz IMyService y un metodo getStatusCode();
6. Registra el servicio en el archivo AndroidManifest.xml

```
public interface IMyService {
  public int getStatusCode();
}
```

LocalService

```
public class LocalService extends Service {

        public void onCreate() {
                super.onCreate();
        }
        public void onDestroy() {
                super.onDestroy();
        }
        public IBinder onBind(Intent arg0) {
                return myBinder;
        }


    public class MyServiceBinder extends Binder implements
IMyService {

        public int getStatusCode() {
                statusCode = 10;
                return statusCode;
        }
    }
}
```

En el manifiesto

```
<service
        android:name=".LocalService">
 </service>
```

Creando un servicio local

En nuestra actividad

```
//En onCreate

Intent intent = new Intent(this,LocalService.class);
        startService(intent);


ServiceConnection conn = new ServiceConnection() {

    public void onServiceConnected(ComponentName name, IBinder service) {
    IMyService myService = (IMyService) service;
    int statusCode = myService.getStatusCode();
    Log.i("MyTag", "Service Bound: Status Code:" +statusCode);

    }

public void onServiceDisconnected(ComponentName arg0) {
// TODO Auto-generated method stub

}
    };

    Intent intent = new Intent(this, LocalService.class);
    bindService(intent, conn, BIND_AUTO_CREATE);
```

Acceso a sensores

Hay muchos sensores:
TYPE_GYROSCOPE
TYPE_PRESURE
TYPR_TEMPERATURE

1. SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
2. Sensor sensor = sensorManager.getDefaultSensor(*Sensor.TYPE_ORIENTATION*);

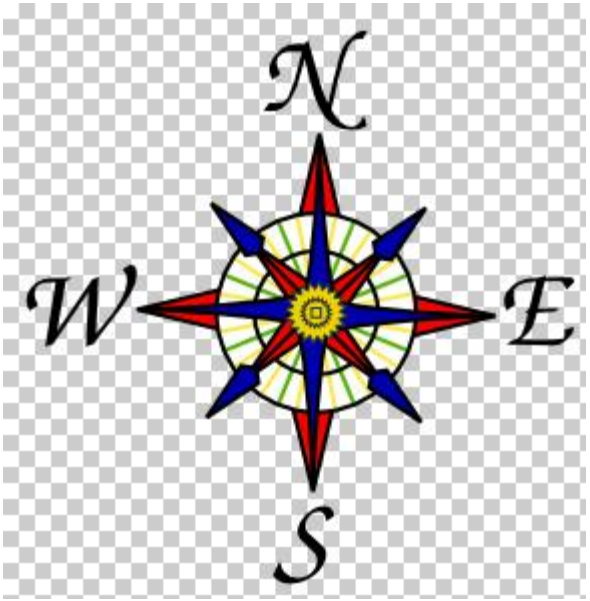
3. Nos tenemos que registrar para escuchar sensores: Hacemos que nuestra Activity implemente a SensorEventListener y rellenamos los métodos generados
4. onSensorChanged // onConfigurationChanged //onResume //onPause

```
// Register to listen to sensors
@Override
public void onResume() {
super.onResume();
sensorManager.registerListener(this, sensor,
SensorManager.SENSOR_DELAY_NORMAL);
}
// Unregister the sensor listener
@Override
public void onPause() {
super.onPause();
sensorManager.unregisterListener(this);
}
```

```
// Escuchamos el sensor
public void onSensorChanged(SensorEvent event) {
int orientation = (int) event.values[0];
Log.d("Compass", "Got sensor event: " + event.values[0]);
rose.setDirection(orientation);
}
```

Acceso a sensores

Ahora nos bajamos una imagen de una brújula de Google imágenes y la guardamos en res/drawable-hdpi



Acceso a sensores

```
public class Rose extends ImageView {
int direction = 0;
public Rose(Context context) {
super(context);
this.setImageResource(R.drawable.compas);
}

// Se llama a este método automáticamente para refrescar la img
public void onDraw(Canvas canvas) { //
int height = this.getHeight(); //
int width = this.getWidth();
canvas.rotate(direction, width / 2, height / 2); //
super.onDraw(canvas); //
}

// Llamamos a este método desde nuestra actividad principal
public void setDirection(int direction) {

this.direction = direction;
this.invalidate(); // lo marca para volver a llamar a onDraw
}
}
```

Acceso a la cámara de fotos

¿Para qué nos va a servir?

1. Lectura de códigos 2D
2. Captura, transformación, envío de Fotos
3. Funciones de Realidad Aumentada y realidad mixta

- Dos formas:
  1. Utilizamos una aplicación instalada en nuestro terminal para realizar la foto y que nos la envíe
  2. Nos programamos nuestro visor de fotos en nuestra aplicación (Complicadillo, no lo vamos a tratar)

Acceso a la cámara de fotos

Forma 1: Mediante intents

Vamos a crear un intent para comunicarnos con la cámara de fotos

Tendremos que crear código para manejar la imágen una vez tomada la foto cuando se vuelva a nuestra actividad

```
Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(takePictureIntent, 11);
//codigo q queramos
```

Versus

```
case 5:
    intent = new Intent("android.media.action.IMAGE_CAPTURE");
    break;
```

Acceso a la cámara de fotos

Forma 1: Mediante intents

Vamos a crear un intent para comunicarnos con la cámara de fotos

Tendremos que crear código para manejar la imágen una vez tomada la foto cuando se vuelva a nuestra actividad

```
Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(takePictureIntent, 11);
//codigo q queramos
```

Versus

```
case 5:
    intent = new Intent("android.media.action.IMAGE_CAPTURE");
    break;
```

Acceso a la cámara de fotos

Forma 1: Mediante intents

Vamos a crear un intent para comunicarnos con la cámara de fotos

Tendremos que crear código para manejar la imágen una vez tomada la foto cuando se vuelva a nuestra actividad

public static final String **ACTION_IMAGE_CAPTURE** Since: API Level 3

Standard Intent action that can be sent to have the camera application capture an image and return it.

The caller may pass an extra EXTRA_OUTPUT to control where this image will be written. If the EXTRA_OUTPUT is not present, then a small sized image is returned as a Bitmap object in the extra field. This is useful for applications that only need a small image. If the EXTRA_OUTPUT is present, then the full-sized image will be written to the Uri value of EXTRA_OUTPUT.

See Also

EXTRA_OUTPUT

Constant Value: "android.media.action.IMAGE_CAPTURE"

Acceso a la cámara de fotos

Forma 1: Mediante intents

Vamos a crear un intent para comunicarnos con la cámara de fotos
Tendremos que crear código para manejar la imágen una vez tomada la foto cuando se vuelva a nuestra actividad

```
Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(takePictureIntent, 11);
//codigo q queramos
```

Versus

```
intent = new Intent(Intent.ACTION_DIAL,
        Uri.parse("tel:(+34)616479034"));
startActivity(intent);
```

Acceso a la cámara de fotos

Si nos vamos a la información de la clase MediaStore, nos dice que es un contenedor para las funciones de:

- Imágenes
- Audio
- Video
- Almacenamiento multimedia

ummary

Nested Classes		
class	MediaStore.Audio	Container for all audio content.
class	MediaStore.Files	Media provider table containing an index of all files in the media storage, including non-media files.
class	MediaStore.Images	Contains meta data for all available images.
interface	MediaStore.MediaColumns	Common fields for most MediaProvider tables
class	MediaStore.Video	

Acceso a la cámara de fotos

Forma 1: Mediante intents

Vamos a crear un intent para comunicarnos con la cámara de fotos

Tendremos que crear código para manejar la imágen una vez tomada la foto cuando se vuelva a nuestra actividad

```
//En onCreate()
mImageView = (ImageView) findViewById(R.id.imageView1);
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(takePictureIntent, 11);  //codigo q queramos

protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if (requestCode == 11) {
                Bitmap photo = (Bitmap) data.getExtras().get("data");
                mImageView.setImageBitmap(photo);
        }
    }
```

Ejercicio:

Tenemos que crear un imageView en el layout

Aunque aún no lo hemos dado en clase vamos a intentarlo

Acceso a la cámara de fotos

Modificando el Manifest.xml

Recuerda permisos!

android.permission.CAMERA

```
<uses-feature android:name="android.hardware.camera" />

 <uses-feature
android:name="android.hardware.camera.autofocus"
android:required="false" />
```

http://developer.android.com/guide/topics/manifest/uses-feature-element.html

Acceso a la cámara de fotos

Forma 1: Mediante intents

Mejoras:

1. La cámara está disponible?
2. No comprobamos el requestCode y el resultCode?
http://developer.android.com/training/camera/photobasics.html
3. Incluimos un botón para que al pulsar se inicie la cámara

```
isIntentAvailable(this, MediaStore.ACTION_IMAGE_CAPTURE)
```

```
public static boolean isIntentAvailable(Context context, String action) {
        final PackageManager packageManager = context.getPackageManager();
        final Intent intent = new Intent(action);
        List<ResolveInfo> list =
                packageManager.queryIntentActivities(intent,
PackageManager.MATCH_DEFAULT_ONLY);
        return list.size() > 0;
    }
```

Forma 2: Nos creamos nuestra camara

1. Crea una objeto de tipo Camera
2. Configuramos sus parámetros
3. Vemos previsualización
4. Realizamos la captura
5. Liberamos

Recuerda permisos!

android.permission.CAMERA

```
//Abrir
Camera camera = Camera.open();

//Configurar
Camera.Parameters parameters = camera.getParameters();
parameters.setPictureFormat(PixelFormat.JPEG);
camera.setParameters(parameters);

…

//Liberar camera.release();
```

Acceso a la cámara de fotos

- **Vemos previsualización**

```
camera.setPreviewDisplay(mySurface);

camera.setPreviewCallback(new PreviewCallback() {
            public void onPreviewFrame(byte[] _data, Camera _camera) {
             // TODO Do something with the preview image.
        }
});

camera.startPreview();
 …

camera.stopPreview();
```

Acceso a la cámara de fotos

- Para tomar una foto tenemos que definir objetos de las clases:

- shutterCallback
- **Picturecallback**

```
ShutterCallback shutterCallback = new ShutterCallback() {
    public void onShutter() {
        // TODO Do something when the shutter closes.
    }
};
PictureCallback rawCallback = new PictureCallback() {
    public void onPictureTaken(byte[] _data, Camera _camera) {
        // TODO Do something with the image RAW data.
    }
};
PictureCallback jpegCallback = new PictureCallback() {
    public void onPictureTaken(byte[] _data, Camera _camera) {
        // TODO Do something with the image JPEG data.
    }
};

camera.takePicture(shutterCallback, rawCallback, jpegCallback);
```

Ref: http://marakana.com/forums/android/examples/39.html

Acceso a información de la tarjeta WiFi

- Como para otros sensores tenemos un Manager, en este caso WifiManager, que hay que obtener a través de getSystemService()

- Después se le pregunta al WifiManager por la conexión WiFi actual (devolvera un objeto WifiInfo. Esta información vamos a representarla como un textStatus
- Podemos preguntar al WifiManager por todos las redes disponibles a través de getConfiguredNetworks(). Nos devuelve una lista de WiFiConfigurations

```
 // Setup WiFi
 wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

// Get WiFi status
WifiInfo info = wifi.getConnectionInfo();
textStatus.append("\n\nWiFi Status: " + info.toString());

// List available networks
List<WifiConfiguration> configs = wifi.getConfiguredNetworks();
```

Ref: http://marakana.com/forums/android/examples/40.html

Acceso a información de la tarjeta WiFi

- **Ejercicio: Creamos un nuevo proyecto que muestra por pantalla información recibida sobre las WiFis existentes y calcula cual es la WiFi con la mejor señal**

```
// Setup WiFi
wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

// Get WiFi status
WifiInfo info = wifi.getConnectionInfo();
textStatus.append("\n\nWiFi Status: " + info.toString());

// List available networks
List<WifiConfiguration> configs = wifi.getConfiguredNetworks();
```

- Creamos un broadcastReceiver:

BroadcastReceiver receiver = new WiFiScanReceiver(this);

- Y nos registramos a los eventos de disponibilidad de escaneo (¿qué ponemos en ???)

```
registerReceiver(receiver, new IntentFilter(
WifiManager.????));
```

- Recordemos desregistrarnos del receiver en el onStop(); unregisterReceiver(receiver);

- Creamos un button y en su onClick llamamos a `wifi.startScan();`

Acceso a información de la tarjeta WiFi

- Creamos la clase WiFiScanReceiver con ese código:

```
public class WiFiScanReceiver extends BroadcastReceiver {

AndroidWiFiDemoActivity wifiDemo;

  public WiFiScanReceiver(AndroidWiFiDemoActivity wifiDemo) {
    super();
    this.wifiDemo = wifiDemo;
  }

  @Override
  public void onReceive(Context c, Intent intent) {
    List<ScanResult> results = wifiDemo.wifi.getScanResults();
    ScanResult bestSignal = null;
    for (ScanResult result : results) {
      if (bestSignal == null || WifiManager.compareSignalLevel(bestSignal.level, result.level) < 0)
        bestSignal = result;
    }

    String message = String.format("%s networks found. %s is the strongest.",
        results.size(), bestSignal.SSID);
    Toast.makeText(wifiDemo, message, Toast.LENGTH_LONG).show();

    Log.d(TAG, "onReceive() message: " + message);
  }
}
```

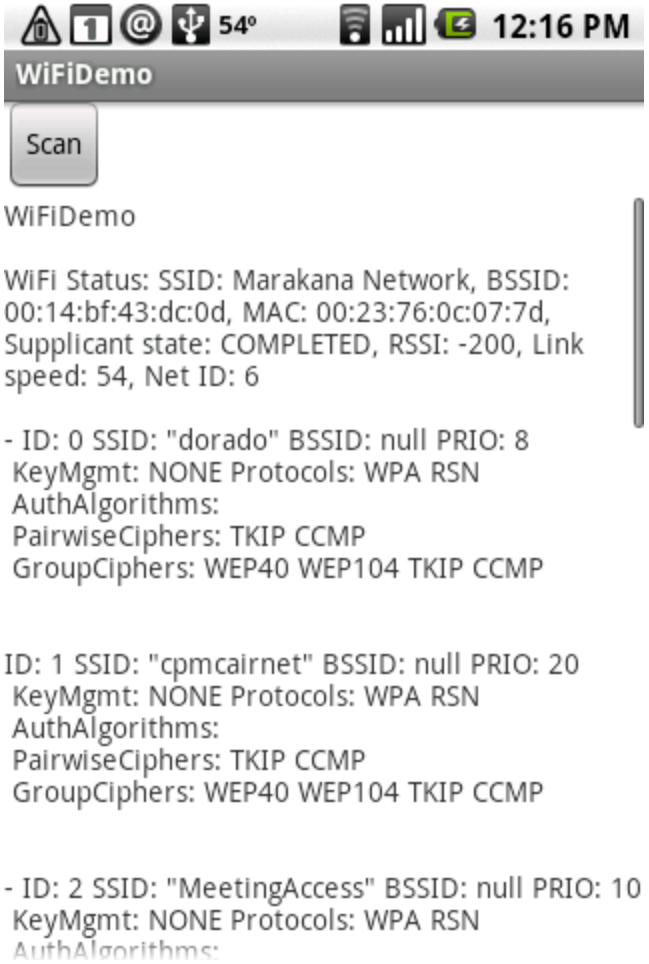
http://developer.android.com/reference/java/util/Formatter.html

Acceso a información de la tarjeta WiFi

- Recordemos crear un Button, un ScrollView con un TextView dentro en el main.xml
- Y poner en el manifiesto:

```
<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
```

- Qué tenemos a la salida?



Ref:
http://marakana.com/forums/android/examples/40.html

- Manejo de notificaciones
  - ¿Qué son las notificaciones?
  - ¿Quién puede producirlas? Actividades y servicios
  - Cuando las utilizamos?

  Generalmente cuando tenemos servicios ya que las actividades sólo pueden realizar acciones cuando están en primer plano.

  - Tipos de notificaciones:
    - De calendario / Email
    - Con mensaje, sonido de alarma, vibración, luz del flash

Ref:
http://developer.android.com/guide/topics/ui/notifiers/notifications.html

- Proceso para crear una notificación
  - Le pedimos al Contexto el NotificationManager:
  - `NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);`
  - Creamos una notificación: `Notification notification = new Notification(…,…,…);`
    - Parámetros:
    - icon: The resource id of the icon to put in the status bar.
    - tickerText: The text that flows by in the status bar when the notification first activates.
    - When: The time to show in the time field. In the System.currentTimeMillis timebase.
  - Creamos un pending intent: ¿Comooorr…?
  - Diferencia entre un pendingIntent y un Intent:
    - Intent: Le digo a una aplicación externa que ejecute un código con sus propios permisos.
    - PendingIntent: Le digo a una aplicación externa que ejecute un código pero con mis permisos (bajo mi responsabilidad)

```
Intent notificationIntent = new Intent(this, NotificationReceiver.class);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);
```

  - Definimos el mensaje de notificación:

```
notification.setLatestEventInfo(this, contentTitle, contentText, pendingIntent);
```

Por último le pasamos la notificación al notification manager:

```
mNotificationManager.notify(int id, notification);
```

- Proceso para crear una notificación

Ejercicio

- Creamos un nuevo proyecto para probar las notificaciones
- Seguimos todo el proceso de la transparencia anterior para crear una notificación.
- Nos bajamos un icono bonito
- Creamos otra actividad llamada NotificationReceiver.java

- Probamos si al ejecutar la clase principal se crea una notificación y si al hacerle click nos lleva a la clase NotificationReceiver que hemos creado

Ref:
http://developer.android.com/guide/topics/ui/notifiers/notifications.html

- Posicionamiento

- Permite determinar donde estamos:
  - Servicios "geocontextualizados"
  - Combinación de cámara y posición para realidad aumentada
  - http://www.xatakamovil.com/aplicaciones/layar-primer-navegador-android-con-realidad-aumentada

  - Recuerda permisos:
    - android.permision.ACCESS_FINE_LOCATION (Para GPS)
    - android.permision.ACCESS_COARSE_LOCATION (Para WiFi, Cell-ID)

- Clases clave: LocationManager, (a través del getSystemService)
  - Permite acceso de servicio de localización
  - Permite obtener los proveedores disponibles

- Posicionamiento

- LocationProvider:
  - Todos los proveedores de localización (incluso podríamos crear uno!)
  - Los más usados: LocationManager.GPS_PROVIDER
  - LocationManager.NETWORK_PROVIDER
  - Tenemos el método: getBestProvider(criteria, enabledOnly)

- Métodos interesantes de LocationManager
  - getLastKnownLocation (No tiene por qué ser la actual)
  - addGpsStatusListener (Nos subscribimos a cambios del GPS)
  - addProximityAlert (Alertas de proximidad: Posición + Radio)
  - requestLocationUpdates (Nos avisa cada vez que cambia la posición)

- Posicionamiento

- requestLocationUpdates:
  - Necesita 4 parámetros:
    - Proveedor a utilizar
    - Tiempo mínimo entre actualizaciones (en milisegundos)
    - Movimiento mínimo entre actualizaciones (metros)
    - LocationListener donde recibiremos las notificaciones

- LocationListener
  - Java Interface, utilizada parar recibir notificaciones del LocationManager
  - Al implementarla nos obligará a implementar sus 4 métodos:
    - onLocationChanged(Location location)Called when the location has changed.
    - onProviderDisabled(String provider)Called when the provider is disabled by the user.
    - onProviderEnabled(String provider)Called when the provider is enabled by the user.
    - onStatusChanged(String provider, int status, Bundle extras)Called when the provider status changes.

- Posicionamiento

- Ejercicio:
  - Aplicación que te muestre todos los proveedores de localización y la información de precisión, soporte de altitud y consumo de recursos para cada uno de ellos.
  - Elegimos uno de ellos (GPS), le preguntamos al location manager si está activado.
  - Creamos tres botones, uno para registrar el listener (requestLocationUpdates), otro para desregistrarlo (removeUpdates) y el tercero nos ofrece un Toast con la última localización
  - Por último imprimimos en pantalla o en trazas cualquier cambio de localización (hay que rellenar el onLocationChanged)

Ref:
http://developer.android.com/reference/android/location/LocationManager.html

- Posicionamiento

- Buenas prácticas:
  - Registrarnos con el locationManager() en el onResume()
  - Desregistrarnos en el onPause()
  - Así cuando la aplicación está en segundo plano el servicio de localización con consume batería.

Ref:
http://developer.android.com/reference/android/location/LocationManager.html

- GeoCodificación
  - Transformar posiciones GPS en direcciones
  - Clase Geocoder(Context), tiene dos métodos principales

  - Puede ser directa:
    - Obtener posición desde una dirección
    - getFromLocationName(address, maxResults)

  - O inversa:
    - Obtener una dirección desde una posición
    - getFromLocation(latitude, longitude, maxResults)

  - Geocoder requiere Internet:

    android.permission.INTERNET

Ref:
http://developer.android.com/reference/android/location/Geocoder.html

- GeoCodificación
  - getFromLocationName(address, maxResults)
  - getFromLocation(latitude, longitude, maxResults)

- Ejercicio:
  - Modificamos la práctica anterior para realizar una GeoCodificación inversa de la localización que obtuvimos con getLastKnownLocation
  - Pista: la geolocalización inversa nos va a devolver una lista de direcciones:
    - List<Address> addresses
  - Luego tenemos que recorrerla (con un for) y extraer las direcciones.
  - Una vez que tenemos las direcciones vamos a hacer un Intent para representar la primera dirección en un mapa, como hicimos en la práctica de los Intents implícitos.
    - PISTA: Para obtener una String con la dirección invocamos getAddressLine(0) del objeto address.

Ref:
http://developer.android.com/reference/android/location/Geocoder.html

- Bluetooth
  - Pedimos el adaptador bluetooth mediante:

BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();

Para empezar el descubrimiento Bluetooth:
adapter.startDiscovery();

Si tenemos dispositivos emparejados con el nuestro, los mostramos con el comando:
Set<BluetoothDevice> devices = adapter.getBondedDevices();

Como siempre importante los permisos:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Ref:
http://marakana.com/forums/android/examples/50.html

- Preferencias
  - Datos que guarda una aplicación para personalizar la experiencia de usuario: Información personal, opciones de presentación.
  - Las preferencias se podrían guardar en la base de datos SQLite de Android pero existe otro método específico para guardar preferencias: preferencias compartidas
  - Preferencias, compuestas de la forma clave-valor. Por ejemplo:

email=ralcarria@dit.upm.es

  - Las prefs se guardan en ficheros XML, en concreto en la siguiente dirección del teléfono móvil:

/data/data/<nombre_del_paquete>/shared_prefs/<nombre_del_archivo>

  - Para ver este archivo abrimos la perspectiva DDMS y lo buscamos
  - Solo funciona en el emulador!

- Preferencias
  - PreferenceScreen: Describe la pantalla de las preferencias
  - PreferenceCategory: Para dividir en categorías
  - Preference: Que pueden ser de varios tipos:

Tipo	Descripción
CheckBoxPreference	Marca seleccionable.
EditTextPreference	Cadena simple de texto.
ListPreference	Lista de valores seleccionables (exclusiva).
MultiSelectListPreference	Lista de valores seleccionables (múltiple).

- Preferencias
  - Se utiliza un objeto SharedPreferences para guardar y leer preferencias

```
SharedPreferences prefs =
     getSharedPreferences("MisPreferencias",Context.MODE PRIVATE);

SharedPreferences.Editor editor = prefs.edit();
editor.putString("email", "modificado@email.com");
editor.putString("nombre", "Prueba");
editor.commit();
```

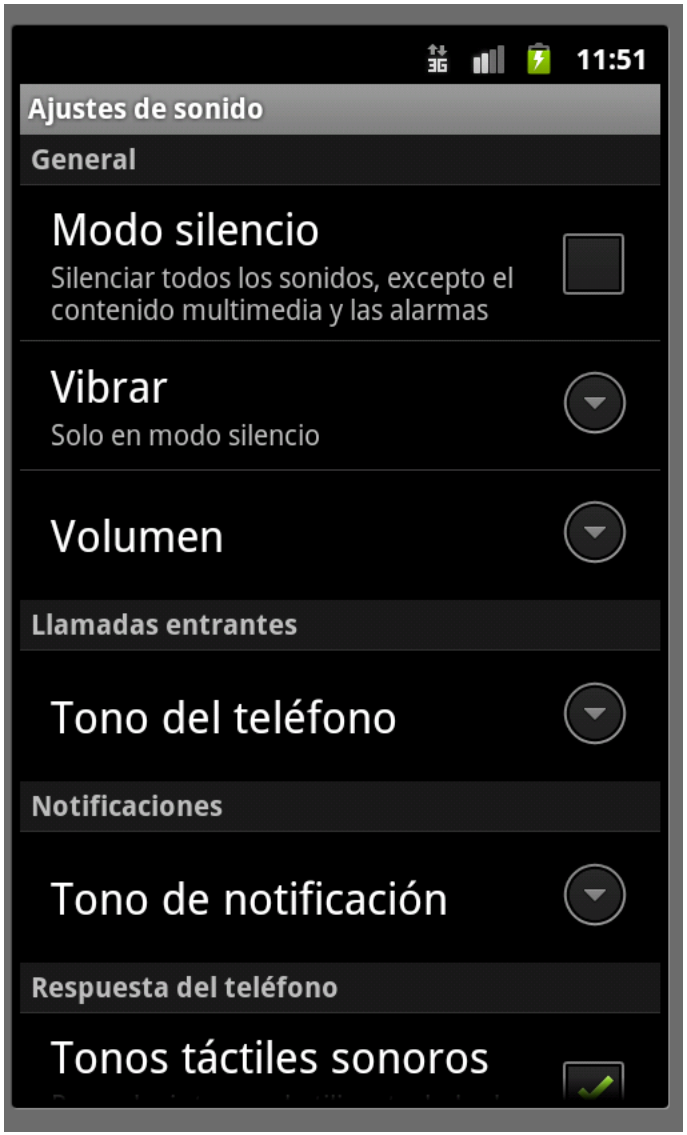
```
String correo = prefs.getString("email", "por_defecto@email.com");
```

  - Vamos a probar a escribir preferencias y luego obtenerlas con estos dos métodos.
  - Vamos a buscar el archivo con el DDMS a ver cual es su aspecto.

- Preferencias
  - Se utiliza un objeto SharedPreferences para guardar y leer preferencias
  - Vamos a probar a escribir preferencias y luego obtenerlas con estos dos métodos.
  - Vamos a buscar el archivo con el DDMS a ver cual es su aspecto.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="nombre">prueba</string>
    <string name="email">modificado@email.com</string>
</map>
```

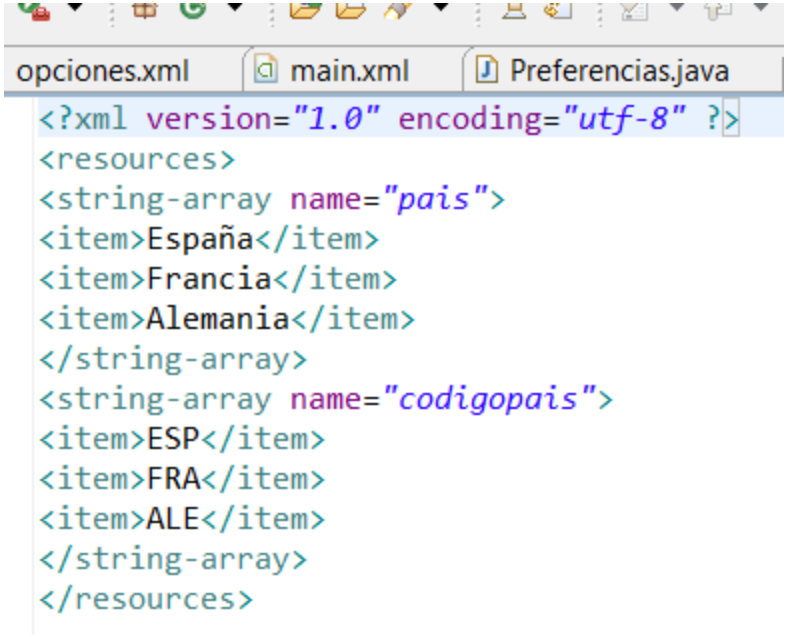
- Preferencias avanzadas
  - Nosotros vamos a trabajar con preferencias sobre el layout de android



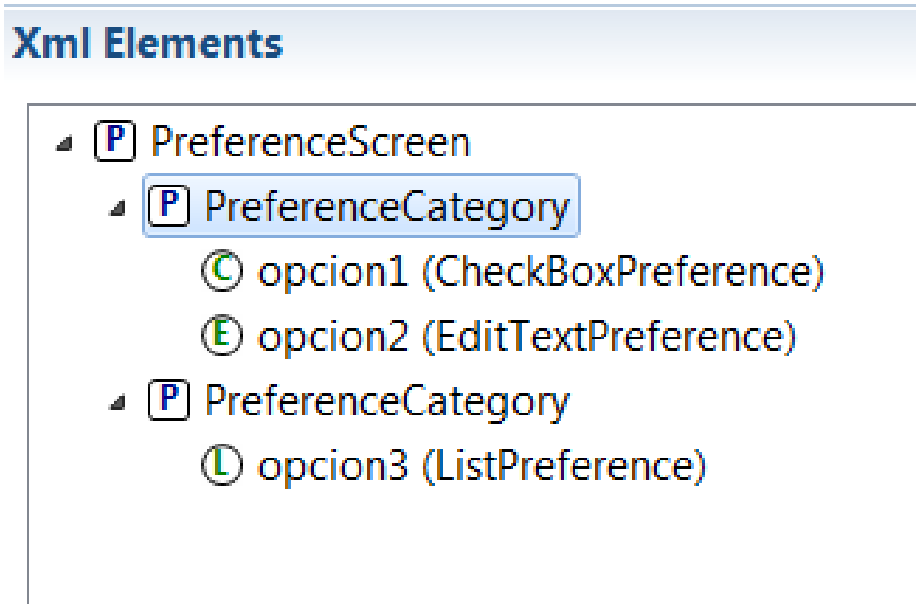
- Practica
  - Creamos una página de preferencias como esta:

- Practica
  - Para ello, en un proyecto nuevo, creamos el archivo paises.xml en /res/values
  - Dentro del archivo creamos un stringArray de nombre “pais” y dentro creamos los items: “España” y “Francia”
  - Cdreamos otro stringArray de nombre “codigopais” y le introducimos los items: “ESP” y “FRA”

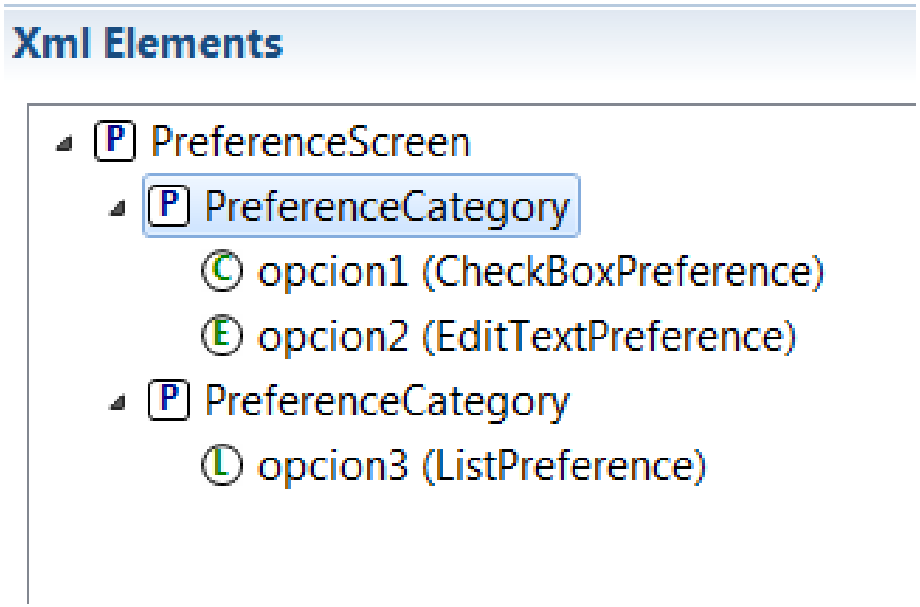
- Practica
  - Para ello, en un proyecto nuevo, creamos el archivo paises.xml en /res/values
  - Dentro del archivo creamos un stringArray de nombre “pais” y dentro creamos los items: “España” y “Francia”
  - Cdreamos otro stringArray de nombre “codigopais” y le introducimos los items: “ESP” y “FRA”



- Practica
  - En /res/xml creamos una archivo “opciones.xml”
  - Le añadimos un “PreferenceScreen” y dentro de este 2 “preference category” y le damos los nombres Categoria1 y Categoria2.



- Practica
  - En /res/xml creamos una archivo “opciones.xml”
  - Le añadimos un “PreferenceScreen” y dentro de este 2 “preference category” y le damos los nombres Categoria1 y Categoria2.



- Practica
  - Rellenamos el resto de información como en este texto

opciones.xml | main.xml | Preferencias.java | AndroidPreferenciasActiv

```
<?xml version="1.0" encoding="utf-8" ?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
<PreferenceCategory android:title="Categoría 1">
<CheckBoxPreference
android:key="opcion1"
android:title="Preferencia 1"
android:summary="Descripción de la preferencia 1" />
<EditTextPreference
android:key="opcion2"
android:title="Preferencia 2"
android:summary="Descripción de la preferencia 2"
android:dialogTitle="Introduce valor" />
</PreferenceCategory>
<PreferenceCategory android:title="Categoría 2">
<ListPreference
android:key="opcion3"
android:title="Preferencia 3"
android:summary="Descripción de la preferencia 3"
android:dialogTitle="Indicar Pais"
android:entries="@array/pais"
android:entryValues="@array/codigopais" />
</PreferenceCategory>
</PreferenceScreen>
```

- Practica
  - Creamos una clase con el siguiente contenido

```
package com.examples;

import android.os.Bundle;

public class Preferencias extends PreferenceActivity{

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        addPreferencesFromResource(R.xml.opciones);
    }
}
```

- Ahora, en nuestra actividad principal creamos un boton (en el main.xml) que al pulsarlo nos vayamos (mediante un Intent) a la actividad PantallaOpciones.
- Recuerda registrar Preferencias en el manifiesto